The version control service for the ATLAS data acquisition configuration files

Igor Soloviev[1,2]

CERN, Geneva 23, CH-1211, Switzerland
[1] UCI University of California, Irvine - Irvine, CA 92697, US
[2] PNPI Petersburg Nuclear Physics Institute, Gatchina, 188300, Russia

Email: Igor.Soloviev@cern.ch

Abstract. The ATLAS experiment at the LHC in Geneva uses a complex and highly distributed Trigger and Data Acquisition system, involving a very large number of computing nodes and custom modules. The configuration of the system is specified by schema and data in more than 1000 XML files, with various experts responsible for updating the files associated with their components. Maintaining an error free and consistent set of XML files proved a major challenge. Therefore a special service was implemented; to validate any modifications; to check the authorization of anyone trying to modify a file; to record who had made changes, plus when and why; and to provide tools to compare different versions of files and to go back to earlier versions if required. This paper provides details of the implementation and exploitation experience, that may be interesting for other applications using many human-readable files maintained by different people, where consistency of the files and traceability of modifications are key requirements.

1. Introduction
The ATLAS experiment [1] is one of the two large general-purpose detectors installed at the Large Hadron Collider (LHC) at CERN. The experiment relies on a complex and highly distributed Trigger and Data Acquisition (TDAQ) system. The configuration service [2] supported by the TDAQ Controls and Configuration working group provides a description of the data taking parameters. They include the description of about 30,000 processes distributed on more than 2,000 nodes with details of their control, data-flow and monitoring configurations, as well as connectivity and parameters for various TDAQ and detector modules, chips and channels. The parameters are stored in XML files prepared and updated by many groups of experts from various TDAQ and detector groups.

The consistency of the XML files and references between data stored in these files is one of the key requirements for reliable and effective functionality of the TDAQ system and the whole ATLAS experiment. This paper explains details of the service developed to control consistency of the files, to check user access permissions and to keep track of changes.

2. Problems of previous implementation
The XML files are stored in a repository located on a shared file system and containing many directories associated with various system or detector components. Until 2008 the files were directly updatable by experts for the different components and the write access to the files was managed by

UNIX group permissions. A special cron job performed backup of files every few hours. This approach had several disadvantages:

- The use of UNIX groups to set access permissions was not sufficiently flexible and difficult to maintain, since in extreme case it required a dedicated group for every directory and assignment of an expert to many groups, that, in turn, had limits from the operating system.
- On various occasions experts left XML files with syntax errors after modifications with text editors, or made changes of a subset of data resulting in dangling references in a complete configuration or even unintentionally removed some files. Sometimes this happened anonymously. The periodic backups allowed the XML files to be restored to an earlier state, but often losing some valid changes.

3. Requirements and choice of new implementation

In the middle of 2008 it was decided that a better way to control, validate and keep track of changes in the XML files was required:

- The files stored in the repository have to be consistent at any given moment, i.e. they may not have syntax errors or dangling references to other files and data.
- It has to be possible to know who made modifications in a file and when, and to access any previous version of a file to see the differences or to recover changes.
- The Access Manager tool [3] responsible for ATLAS users authorization has to be used to check update permissions on files in the repository.

From the above requirements it was clear that direct write access to the repository by experts had to be revoked. Instead an update of the repository should be performed from a single point by a process running under a privileged account on a dedicated node. Two main ideas were considered:

1. To develop a permanently running server application validating changes made by users.
2. To develop a utility validating the changes and applying them to the repository.

After discussions inside the TDAQ community it was decided to implement the second idea: as even though the first one can potentially provide faster response preloading all XML files, it is error prone and requires more development and support.

To provide version control of every file modification it was decided to use the popular Concurrent Versions System (CVS) [4].

4. Details of implementation

The new implementation is called the OKS Server[1]. Its architecture is shown in figure 1.

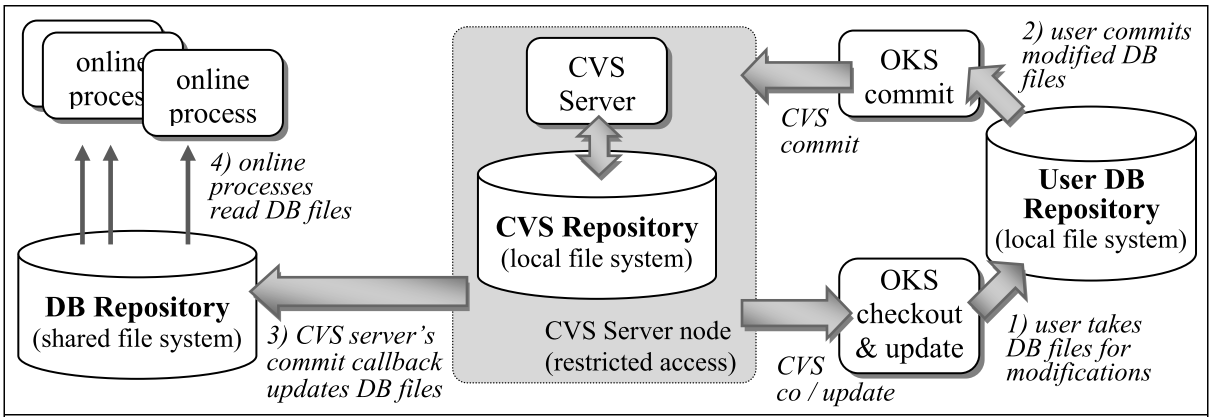


Figure 1. The OKS Server architecture.

[1] Note that OKS (Object Kernel Support) [5] is the run-time object manager package, developed for ATLAS to handle configuration objects, using an object data model. It stores database schema and data files in XML format and can merge the data into a single database in memory. The OKS Server handles the XML files used by OKS.

The OKS Server is based on a CVS server. The user creates or checks out files into a private database repository, makes modifications and commits the files into CVS (using a special OKS commit script). Then the CVS server's callback updates the main database repository. The CVS server runs on a dedicated node (accessible by administrators only) under a privileged account allowing modification of DB repository files. The file system volume storing the OKS DB repository is mounted read-write on the CVS server node and is mounted read-only on all other nodes.

Internally the OKS commit script invokes a special OKS commit binary that checks for OKS consistency and the user's authorization via the Access Manager service. If the files consistency check is passed and the user is authorized for changes of these files, the OKS commit binary performs the necessary steps for CVS user authentication and passes updates to the CVS server. The OKS commit binary has an execution sticky bit so that it runs under a privileged account able to get the CVS server commit authorization, to protect the OKS server from CVS commits bypassing OKS validation.

5. Interfaces

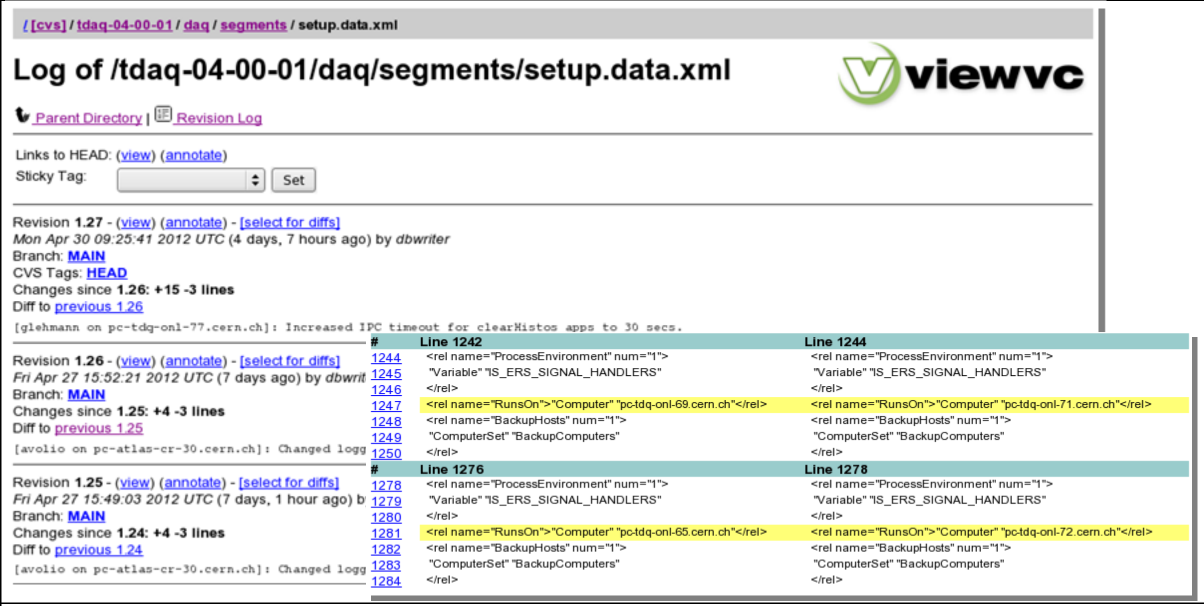
The application programming interface (API) to the OKS Server is integrated into all configuration service packages and is transparent to all applications reading or modifying the database. To update data stored on the OKS Server one can use a native C++ OKS API, as well as the configuration service APIs (C++, Java and Python).

The user programming interfaces include three groups of tools such as command line utilities, graphical editors and read-only Web interface to the OKS server archive.

The command line utilities are similar to CVS ones: a user can check out XML files into a private database repository. The files in the user repository can be modified using any tools including generic text editors. Then the updated files can be committed on the OKS server using the OKS commit script, as well as updated from the OKS Server or released.

The OKS data editor graphical application allows the user to explicitly check out, update, commit and release the OKS Server files. The new database editor [6] based on the configuration service API completely hides from the user the difference between directly writable and OKS Server files.

The Web interface is based on the ViewVC CVS browser (http://www.viewvc.org) and it provides tools to see who made changes in OKS Server files, plus when and why the changes were made. In addition any previous version of a file can be downloaded or differences between versions visualized. An example of the Web interface is shown in figure 2.



/ [cvs] / tdaq-04-00-01 / daq / segments / setup.data.xml

Log of /tdaq-04-00-01/daq/segments/setup.data.xml

Links to HEAD: (view) (annotate)
Sticky Tag:

Revision **1.27** - (view) (annotate) - [select for diffs]
Mon Apr 30 09:25:41 2012 UTC (4 days, 7 hours ago) by *dbwriter*
Branch: **MAIN**
CVS Tags: **HEAD**
Changes since **1.26: +15 -3 lines**
Diff to previous 1.26

```
[glehmann on pc-tdq-onl-77.cern.ch]: Increased IPC timeout for clearHistos apps to 30 secs.
```

Revision **1.26** - (view) (annotate) - [select for diffs]
Fri Apr 27 15:52:21 2012 UTC (7 days ago) by *dbwrit*
Branch: **MAIN**
Changes since **1.25: +4 -3 lines**
Diff to previous 1.25

```
[avolio on pc-atlas-cr-30.cern.ch]: Changed logg
```

Revision **1.25** - (view) (annotate) - [select for diffs]
Fri Apr 27 15:49:03 2012 UTC (7 days, 1 hour ago) b
Branch: **MAIN**
Changes since **1.24: +4 -3 lines**
Diff to previous 1.24

```
[avolio on pc-atlas-cr-30.cern.ch]: Changed logg
```

#	Line 1242	Line 1244
1244	<rel name="ProcessEnvironment" num="1">	<rel name="ProcessEnvironment" num="1">
1245	"Variable" "IS_ERS_SIGNAL_HANDLERS"	"Variable" "IS_ERS_SIGNAL_HANDLERS"
1246	</rel>	</rel>
1247	<rel name="RunsOn">"Computer" "pc-tdq-onl-69.cern.ch"</rel>	<rel name="RunsOn">"Computer" "pc-tdq-onl-71.cern.ch"</rel>
1248	<rel name="BackupHosts" num="1">	<rel name="BackupHosts" num="1">
1249	"ComputerSet" "BackupComputers"	"ComputerSet" "BackupComputers"
1250	</rel>	</rel>
#	**Line 1276**	**Line 1278**
1278	<rel name="ProcessEnvironment" num="1">	<rel name="ProcessEnvironment" num="1">
1279	"Variable" "IS_ERS_SIGNAL_HANDLERS"	"Variable" "IS_ERS_SIGNAL_HANDLERS"
1280	</rel>	</rel>
1281	<rel name="RunsOn">"Computer" "pc-tdq-onl-65.cern.ch"</rel>	<rel name="RunsOn">"Computer" "pc-tdq-onl-72.cern.ch"</rel>
1282	<rel name="BackupHosts" num="1">	<rel name="BackupHosts" num="1">
1283	"ComputerSet" "BackupComputers"	"ComputerSet" "BackupComputers"
1284	</rel>	</rel>

Figure 2. Example of the OKS Server web archive interface.

6. Experience in ATLAS

The OKS Server was used in ATLAS since late 2008, including all of the subsequent LHC data taking. During this period there were no errors of the server itself and no problems with data inconsistency in the files stored on the OKS Server. Thus proving the robustness and security of the chosen approach and justifying the necessity of the version control service for the ATLAS TDAQ configuration files.

At present the OKS Server manages more than 1300 XML files for the ATLAS data taking configuration and their total size exceeds 160 Mbytes. The overhead introduced by the OKS Server to commit changes in the configuration is typically less than 5-10 seconds including validation of changes, commit to the CVS server, and updating of the files on the shared file system by the CVS server callback.

This or a similar technology would be equally advantageous wherever human readable data files are writable by several experts and their consistency and traceability of modifications are key requirements for functionality of a system.

Acknowledgments

The author wishes to acknowledge the encouragement and helpful feedback during the design and prototyping stages of the OKS Server from colleagues and the special work on the CVS server and file system setups by ATLAS TDAQ system administrators.

References

[1] ATLAS Collaboration 2008 “The ATLAS Experiment at the CERN Large Hadron Collider” *JINST* 3 **S08**003

[2] Soloviev I et al. 2007 "The ATLAS DAQ system online configurations database service challenge" *J. Phys.: Conf. Ser.* **119**:022004

[3] Leahu M, Dobson M and Avolio G 2008 “Access Control Design and Implementations in the ATLAS Experiment” *IEEE Trans.Nucl.Sci.* **55**-1, 386-391

[4] “Essential CVS” 2006 ISBN: 0596527039

[5] Soloviev I, Jones R, Mapelli L and Ryabov Y 1998 “The OKS persistent in-memory object manager” *IEEE Trans.Nucl.Sci.* **45**-4-1, 1958-1964

[6] Bianchi R et al. 2011 “Control and configuration of the ATLAS trigger and data acquisition system during data taking activities“ PoS(EPS-HEP2011)486